AFRL-RI-RS-TR-2009-182
Final Technical Report
July 2009

WIRELESS COMPUTING ARCHITECTURE

Harvard University

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report was cleared for public release by the 88th ABW, Wright-Patterson AFB Public Affairs Office and is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2009-182 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/s/
KERRY PAHAL, 2Lt, USAF
Work Unit Manager

/s/
EDWARD J. JONES, Deputy Chief
Advanced Computing Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
JULY 2009	Final	March 2008 – June 2009

4. TITLE AND SUBTITLE
WIRELESS COMPUTING ARCHITECTURE

5a. CONTRACT NUMBER
N/A

5b. GRANT NUMBER
FA8750-08-1-0220

5c. PROGRAM ELEMENT NUMBER
63662D

6. AUTHOR(S)
H.T. Kung and William H. Gates

5d. PROJECT NUMBER
WCNA

5e. TASK NUMBER
WC

5f. WORK UNIT NUMBER
AH

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Harvard University
33 Oxford Street
Cambridge, MA 02138

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AFRL/RITC
525 Brooks Road
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)
N/A

11. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-RI-RS-TR-2009-182

12. DISTRIBUTION AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED. PA# 88ABW-2009-3339 Date Cleared: 22-July-2009

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Work of the project has led to a number of findings and conclusions. These results have been published or will be published in the near future:
-Localization with Snap-Inducing Shaped Residuals (SIRS) – Coping with Errors in Measurement, "15th Annual International Conference on Mobile Computing and Networking (MobiCom 2009), September 2009.
-"A Computational Wireless Network Backplane: Performance in a Distributed Speaker Identification Application," Military Communications Conference (MILCOM 2008), November 2008.
-Rainbow: A Wireless Medium Access Control Using Network Coding for Mulit-hop Content Distribution," Military Communications Conference (MILCOM 2008), November 2008.
Our work has mainly been motivated by the need of providing on-demand local computing resources over a wireless computing and communication infrastructure. In the next decade we will begin to face very large sensor-generated datasets on the order of zettabytes or even yottabytes. While centralized control and applications may still be needed, the bulk of the processing and storage must be distributed near the sensors. Moreover, for flexibility in resource deployment, communications over wireless networks will be essential. Technologies need to be developed which can cope with modest bandwidths of wireless communications while being able to take advantage of their broadcast capabilities. We have obtained new results capable of addressing these challenges. In the rest of this document, we highlight our accomplishments in three related areas.

15. SUBJECT TERMS
Muli-hop Content Distribution, Wireless Medium Access Control, Rainbow, content-directed medium access control, MAC priority scheme, MAC-level bottleneck, Snap-Inducing Shaped Residuals (SISR)

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Kerry J. Pahal
U	U	U	UU	53	19b. TELEPHONE NUMBER (*Include area code*) N/A

TABLE OF CONTENTS

LIST OF FIGURES

1.0 SUMMARY

Work of this project has led to a number of findings and conclusions. These results have been published or will be published in the near future:

- "Localization with Snap-Inducing Shaped Residuals (SISR) - Coping with Errors in Measurement," 15th Annual International Conference on Mobile Computing and Networking (MobiCom 2009), September 2009.
- "A Computational Wireless Network Backplane: Performance in a Distributed Speaker Identification Application," Military Communications Conference (MILCOM 2008), November 2008.
- "Rainbow: A Wireless Medium Access Control Using Network Coding for Multi-hop Content Distribution," Military Communications Conference (MILCOM 2008), November 2008.

Our work has mainly been motivated by the need of providing on-demand local computing resources over a wireless computing and communication infrastructure. In the next decade we will begin to face very large sensor-generated datasets on the order of zettabytes or even yottabytes. While centralized control and applications may still be needed, the bulk of the processing and storage must be distributed near the sensors. Moreover, for flexibility in resource deployment, communications over wireless networks will be essential. Technologies need to be developed which can cope with modest bandwidths of wireless communications while being able to take advantage of their broadcast capabilities. We have obtained new results capable of addressing these challenges. In the rest of this document, we highlight our accomplishments in three related areas.

2.0 INTRODUCTION

2.1 Coping with Errors in Sensor Measurement

We consider the problem of localizing wireless nodes in an outdoor, open-space environment, using ad-hoc radio ranging measurements, e.g., 802.11. As in other range-based methods, we cast ranging measurements as a set of distance constraints, thus forming an over determined system of equations suitable for non-linear least squares optimization. However, ranging measurements are often subject to errors, induced by multipath signals and variations in path loss, or even faulty hardware or antenna connectors. Including such potentially large and non-Gaussian errors in the measurement data ultimately produces inaccurate localization solutions. We have developed a new method, called *snap-inducing shaped residuals* (SISR), to automatically identify “bad nodes” and “bad links” arising from these errors, so that they receive less weight in the localization process. In particular, SISR snaps “good nodes” to their accurate locations and gives less emphasis to other nodes. While the mathematical techniques used by SISR are similar to those in robust statistics, SISR’s exploitation of the snap-in effect in localization appears to be novel. We have provided analysis on the principle of SISR, and demonstrate a working SISR implementation in field experiments on a testbed of 20 wireless nodes, as well as the superior performance of SISR in simulation with a larger number of nodes.

2.2 A Computational Wireless Network Backplane for Distributed Speaker Identification

A major challenge in the DoD's next-generation network-centric information systems concerns on-demand provisioning of computation and network infrastructures at tactical network edges (e.g., deploying wireless airborne or hybrid air/ground networks). To support this vision, we have developed DWARF, a general distributed application execution framework for wireless ad-hoc networks which dynamically allocates computation resources and manages failures. DWARF nodes each run a separate task simultaneously, thereby achieving execution speed-up from parallel processing. Failed tasks, e.g., due to fluctuating wireless links to mobile nodes, are automatically detected and reassigned, transparent to the application. Further, tasks are executed in an order that satisfies dependencies given by task dependency graphs. To use DWARF, application programmers need only decompose their applications into tasks and define the task dependency graphs. In this work, we describe DWARF and report its benefits in running an important existing application— speaker identification—over a 32-node wireless network which supports fault-tolerant computation. We observed two major performance gains: (1) a ten-fold speed-up in identifying speakers due to parallelizing the application, and (2) higher accuracy in speaker identification, made possible by the increased sensor diversity provided by geographically distributed nodes. While our nodes have modest computing power individually, combined under DWARF, they are able to execute speaker identification with much greater speed and with improved accuracy.

2.3 Use of Network Coding for Wireless Multi-hop Content Distribution

We consider the problem of multi-hop content distribution over a wireless ad-hoc network. Such mechanisms are relevant to a broad spectrum of applications, but are particularly important to data broadcast in wireless distributed computing where speedy I/O is critical to overall performance. We have developed Rainbow, a content distribution protocol for multi-hop wireless ad-hoc networks. The protocol uses a content-directed medium access control (MAC), through which transmission priority is given to those nodes most capable of delivering useful content to their neighbors. We have developed an efficient implementation of Rainbow based on network coding. Specifically, Rainbow uses a MAC priority scheme, where the priority of packet transmission from a node depends on the rank of the coefficient matrix associated with the coded content the node holds. We have demonstrated that Rainbow achieves a 1.3- to 1.9-fold improvement in content distribution time over other flooding protocols, as measured on a testbed of 29 wireless nodes. We attribute this performance gain in part to Rainbow's ability to address a MAC-level bottleneck in multi-hop wireless networks, which we refer to as the "bridge lock-out problem".

3.0 METHODS, ASSUMPTIONS, AND PROCEDURES

3.1 Snap-Inducing Shaped Residuals (SISR)

In this section, we present a novel kind of residual function for use in optimization-based localization. We will say that the residual function is *snap-inducing* due to its tendency to preserve residuals smaller than some threshold, while diminishing the effect of the larger ones. We give a theoretical analysis of the SISR estimator and show that the properties of SISR can be explained by the theory of robust statistics.

The localization problem is typically framed as a non-linear least squares optimization, where ranging measurements are used as constraints, and a best fit is sought to minimize squared residuals. In the context of localization, the residual is defined as the difference between pairwise ranging measurements and the distances estimated by the estimator. Equation 1 shows the formulation of the residual r(i,j) from a ranging measurement between nodes i and j

$$r(i,j) = \hat{d}_{ij} - d_{ij} \tag{1}$$

where $\hat{d}_{ij}$ is the distance estimated by the least-squares estimator and d_{ij} is the ranging measurement between the two nodes.

Conventional least squares works by minimizing an objective function, which is the sum of squared residuals over all node pairs (i, j):

$$F = \sum_{i,j} r(i,j)^2 \tag{2}$$

The squared residual function is shown in Figure 1. The least-squares estimator is not robust to extremely noisy measurements because the squared residual grows quadratically. To solve the problem as well as produce the snapped-in effect for good nodes, we propose a new *snap-inducing shaped residual* (SISR) estimator, which has a shaping function that deemphasizes the influence of bad nodes while emphasizing the good ones. The function is sketched in Figure 1 and has the following two properties.

1. The shaping function increases with a smaller slope when the residual is large. In particular, the function dampens the impact of residuals larger than a threshold τ. We call this the *wing-shaped section*.

2. The shaping function has a narrow and deep well for residuals close to 0. The potential good measurements can therefore be emphasized by growing the shaped residuals or the "cost function" more rapidly. This is called the *U-shaped section*.

The general form of the SISR shaping function is shown in Equation 3:

$$s(i,j) = \begin{cases} \alpha r(i,j)^2 & \text{if } |r(i,j)| < \tau \\ \ln(|r(i,j)| - u) - v & \text{otherwise} \end{cases} \tag{3}$$

where α, τ, u, and v are parameters to be configured.

Tuning the shape of the SISR function controls its sensitivity to errors. The parameter α is introduced to control the height of the U-shaped section, while τ controls its width. (Note that the "U" portion has its x-axis values in the range [-τ, +τ].) The wing-shaped section is created by taking logarithms of the residuals larger than τ. Note that in robust statistics literature, a completely flat function is often used to produce similar robust behaviors [1]. However, a flat objective function with zero slope is usually difficult to handle numerically. Numerical algorithms such as Levenberg-Marquardt require a non-zero gradient of the objective function, and therefore we choose a slowly increasing log function instead. Another common requirement for numerical methods is that the objective function has to be continuous and differentiable. To make the SISR function piecewise-continuous and piecewise-differentiable at τ, the other two parameters u and v should have the form shown in Equations 4 and 5, respectively.

$$u = \tau - \frac{1}{2\alpha\tau} \tag{4}$$

$$v = \ln\left(\frac{1}{2\alpha\tau}\right) - \alpha\tau^2 \tag{5}$$

In short, the SISR function is controlled by two tunable parameters α and τ. The SISR estimator operates by minimizing the objective function as the sum of the shaped residuals over all node pairs (i, j).

$$F = \sum_{i,j} s(i,j) \tag{6}$$

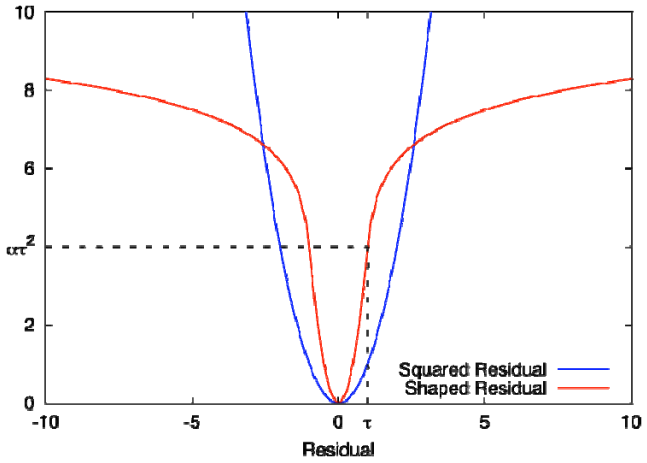
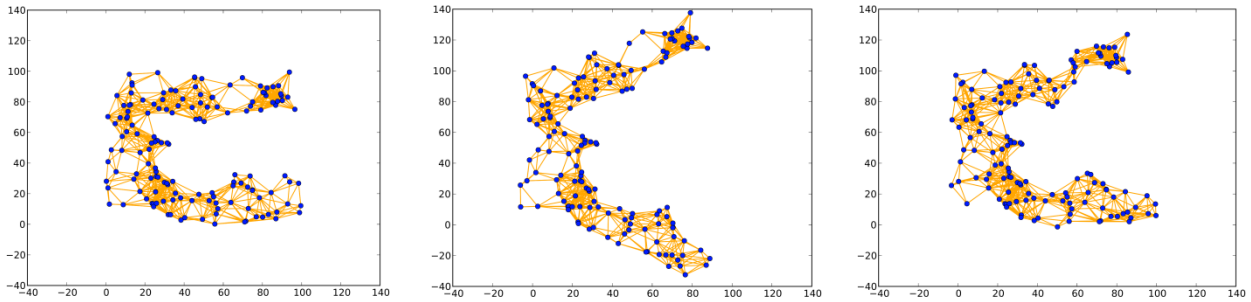


Figure 1: A Comparison between Least Squares Residual and SISR Functions

We compare SISR against an optimized variant of multidimensional scaling – MDS-MAP(C,R) [2]. A key insight in MDS-MAP is the replacement of the ranging measurement matrix with a ``proximity matrix" P, where p_{ij} is the shortest path distance between nodes i and j. This essentially mitigates (the common) ranging error where the distance between two nodes is over-estimated and, in multi-hop topologies, provides a proxy ranging estimate for node pairs that are not within communications range. MDS-MAP(C,R) further develops this insight by performing a weighted least squares refinement on a classic MDS-MAP solution, where constraint weights are inversely proportional to the number of hops in the shortest path. This strategy dampens the effect of long-distance estimates, which are more susceptible to errors, but makes the blanket assumption that long-distance estimates are always less trustworthy. In contrast, SISR can discern between good and bad ranging measurements and automatically use only those that are good.

In general, MDS-MAP is an effective method for multi-hop localization. However, for certain non-convex multi-hop topologies the shortest path distance can give rise to errors in localization, a recognized limitation of such approaches [3], [4]. For example, in the “C”-shaped topology in Figure 2(a), the true distance between nodes at the ends of the “C” is much smaller than the shortest path distance would indicate. On such a topology, MDS-MAP tends to force apart the ends of the “C” (Figure 2(b)). The presence of ranging errors tends to exacerbate the problem. For this reason, we focus on these difficult topologies in our simulations.



(a) (b) (c)

Figure 2: Non-convex Topologies

(a) A non-convex "C"-shaped topology with 147 nodes randomly placed within a 100m x 100m bounding area. Topologies such as these were used as ground truth in simulations described in Section 4.1. (b) The corresponding MDS-MAP(C,R) solution. Deformation in the ends of the "C" are clearly visible. (c) The corresponding SISR solution, which corrects for the deformation, even when ranging errors are present

Given a reasonably good initial approximation, SISR can arrive at a very accurate localization solution for nodes that have good ranging measurements. MDS-MAP(C,R) results are good candidates for initial approximations, and as such, we use them in our simulations.

The most natural way to evaluate the performance of these algorithms is through an *absolute localization error metric*, i.e., the discrepancy between a node's computed location and its ground truth location. However, the formulation of SISR we have presented thus far is an anchor-free, relative localization scheme; it depends only on pairwise distance measurements and does not return its results in the same coordinate system as the ground truth. The core MDS-MAP(C,R) algorithm is also such a scheme.

In order to report absolute localization error, we modified both algorithms slightly to employ the ground truth locations of four anchor nodes. As in [2], the anchors are randomly selected. For MDS-MAP(C,R), the solution returned by classic MDS-MAP is first transformed (via rotation, translation and reflection) into the ground truth coordinate system by fitting the anchors to their ground truth locations. This provides a better-conditioned initial solution to the subsequent refinement stage, where the anchor locations are further used as additional, fixed-constraint equations passed to the weighted least squares process. This forces the least squares operation to respect the ground truth coordinate system. The same technique is employed in providing constraint equations to SISR. Note that the way we use anchor information is different from the method in [2], where the final localization solution is fitted to the ground truth via anchors. Since we postulate the availability of anchor information, it makes sense to take advantage of it during the localization procedure.

In 2D localization, three anchors are sufficient, but their random selection can cause ambiguities during coordinate system fitting (e.g., if the three anchors are almost collinear). In [2], a fourth anchor is used to reduce the ambiguity, but there is no guarantee that this will always work. Thus, a question arises: what criteria describe a good set of anchors? First, they must be well-separated. Second, the angles between any pair must not be near zero. We note that, in its anchor-free, relative localization formulation, SISR can be used to bootstrap an absolute localization by providing hints as to which nodes satisfy these two criteria. First, SISR's snapped-in effect allows accurately localized nodes to be distinguished by their low residuals. Second, from the relative localization solution, we can distinguish which among the low-residual nodes are well-separated. However, for fairness in comparison with MDS-MAP(C,R), we do not employ this method in our evaluations.

3.2 Distributed Wireless Application Runtime Framework (DWARF)

In this section we shift gears and describe DWARF, an execution system built to support distributed applications in 802.11 wireless ad-hoc networks. We consider DWARF to be a *computational wireless network backplane*. Conventional wisdom suggests that backplanes connecting parallel processors are required to have high reliability, bandwidth, and throughput; an 802.11 wireless network could hardly fit the bill. However, DWARF's fault-tolerant design mitigates the unreliability of wireless links and its extensive use of wireless broadcasts allows data to be sent to many nodes in one shot (e.g., in input data distribution), thereby compensating for the relatively low bandwidth of the wireless channel in many situations.

We give a detailed description of DWARF below. Overall, we assume that a node, termed "master," exists at any given time to drive the computation from start to end. This need not be a single physical node, as the participating nodes can monitor the status of the master node and promote a backup node if the master fails.

Note that all wireless transmissions in DWARF are broadcasts; none are point-to-point. We rely on the 802.11 MAC (CSMA) to handle transmission scheduling and channel contention, and assume that all nodes reside in a single collision domain.

3.2.1 Reliable Distribution Input Data

Due to the nature of wireless channels multiple receivers can listen to the same physical transmission, a property referred to herein as *broadcast advantage*. Therefore, we have focused attention on problems where the same input data simultaneously serve a multitude of computation tasks on separate nodes. This is more efficient than repeatedly using the channel to independently transmit the input to each node.

Our framework specifies a reliable broadcast mechanism to distribute input data to the nodes participating in a computation. In our implementation, we use a single-hop reliable broadcast protocol based on acknowledgments and retransmissions, structured as follows. We divide input files of arbitrary size into fixed-size *generations*, each consisting of some number G of link layer packets. The master node then reliably transmits generation by generation until the entire file is transmitted. Within each generation, every T_G seconds the master resends all outstanding packets where outstanding packets are those that haven't been received by at least one node. Nodes acknowledge each received packet, or in the absence of received packets periodically announce the subset of the current generation they hold. This mechanism ensures that the transmission will make progress even in lossy channels, as long as the probability of success is non-zero. This reliable broadcasting mechanism is relatively simple compared to some advanced techniques described in the ad-hoc networking literature [5]. We envision a replacement scheme based on network coding in the future.

3.2.2 Task Decomposition

Our framework expects that applications be divided into units of execution termed “tasks”. The definition of tasks is up to application writers, but usually they are modules with relatively few dependencies that can run concurrently, thereby achieving parallel speed-up.

The dependency relationships between tasks are described by application writers in dependency graphs. A directed edge between nodes A and B in the graph ($A \rightarrow B$) specifies that task A must complete before B starts, for reasons such as task B requiring input from A, or to impose a certain ordering of side-effects.

In our implementation, task inputs and outputs are opaque to the execution system, and are considered to be files of arbitrary format. In the future we will extend this interface to allow formal data types; with this, the system can inspect application results and perform *task pruning*. The output of each task is reliably broadcast to the whole computation group for increased system reliability in the face of node and link failures. Note that since any transmission is likely to be overheard in a wireless medium by nearby nodes, adding a reliability mechanism does not increase the resource demands as much as with point-to-point channels. The reliable output broadcasting mechanism is similar to that used for inputs as described in the previous subsection.

3.2.3 Distributed Fault-tolerant Task Scheduler

In an environment prone to faults, nodes are not guaranteed to finish their tasks within a predetermined time and may even exit the computation abruptly. Fault-tolerant schemes [6], [7], [8] must be used to ensure all tasks eventually complete. While allowing repeated task execution for fault tolerance, such schemes need to minimize unnecessary redundancy in task execution.

DWARF satisfies these requirements by employing an *optimistic scheduler with fully-replicated control structures* at all nodes. These control structures (detailed below) provide each node in the DWARF system with a view of the global system state and, when kept consistent across nodes, allow the scheduler to assign unfinished tasks efficiently, while minimizing task redundancy. Such a strategy is particularly well-suited to the wireless medium because broadcast advantage significantly reduces the overhead of control structure replication.

Based on this replicated control mechanism, the DWARF scheduler tracks node liveness in order to detect and react to node failure and reassign failed tasks. Each DWARF node runs an instance of the scheduler and periodically broadcasts a heartbeat message. Upon receipt of a heartbeat, a DWARF node records the arrival timestamp in a table, with one entry per node. Nodes are marked as "failed" if subsequent heartbeats are not received after a timeout period. If, at a later time, a heartbeat arrives from a failed node, the scheduler will promote the node back to "live" status.

The scheduler manages task execution by maintaining two data structures: a *dependency graph* with all unfinished tasks and a *priority queue* of ready-to-run tasks (i.e., tasks with completely satisfied dependencies). When the scheduler starts, it performs a topological sort of the dependency graph, resulting in tasks with the fewest dependencies being given the highest priority. This maximizes the number of independent tasks available at any given time and minimizes node idle time.

To minimize redundant work, multiple nodes should avoid scheduling and invoking the same task. To do so, each scheduler broadcasts notifications of task scheduling and completion events and monitors which tasks have been scheduled or completed by other nodes, placing tasks currently being executed by any node at the end of the queue and removing them upon completion. If node failure is detected by the scheduler and results in a task being unfinished, that task is moved back into its original position in the queue.

The scheduler makes use of a *distributed advisory locking mechanism* [9] in order to avoid the situation where multiple nodes schedule the same task at exactly the same time. In this mechanism, node IDs are sorted and placed into a virtual ring topology. In an N-node system, the scheduler on each node adheres to the following lock request policy. A scheduler requests a lock for task t from node i, where i = t mod N. If node i has failed, a new request is made to node (i+1) mod N instead. If this too has failed, (i+2) mod N is tried, and so on, until a live node is found. Any scheduler receiving a lock request for task t will grant it, provided that it has no outstanding locks on t and has not recently heard another scheduler grant a lock on t. Once granted, a lock must be refreshed periodically as it is automatically released after a certain period of time q. This prevents deadlock when a node holding a lock fails.

Note that each node in the system updates its own lock table when it overhears a lock being granted by another node. As a result, any given node has the most complete picture of current system state possible in the presence of faults. In the case where inconsistencies in notions of node liveness exist, multiple locks for the same task may still be granted by different maintainers, resulting in some redundancy in task execution. However, this is kept to a minimum by the control structure replication.

The fault-tolerant task scheduler algorithm running on each DWARF node is summarized in Figure 3. On invocation, the scheduler initializes its control structures, and immediately enters the main loop. First, the scheduler checks if all tasks are finished. If so, then all computation is done and the scheduler exits. If not, the scheduler pops the highest priority task from its task queue, immediately begins execution and then sends a lock request. Note that this is why DWARF's scheduler is *optimistic* -- it does not wait for a task lock before beginning execution. During execution, the scheduler keeps track of the task's lock status. If the lock is granted, a timer on the lock is started. When the lock is about to expire, but the task has yet to finish, the scheduler refreshes the lock by sending another lock request. If a lock request is eventually refused, the scheduler terminates the task, pushes it onto the tail of the task queue, and returns to the beginning of the main loop. When a task completes, the scheduler broadcasts a completion notification and returns to the beginning of the main loop. On receipt of a task completion notification, the scheduler removes the task from its own task queue.

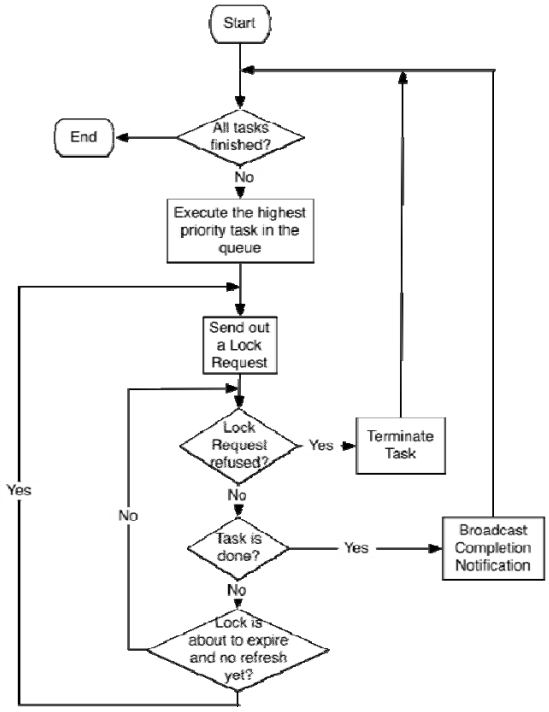


Figure 3: The DWARF Fault-tolerant Task Scheduler Algorithm

3.2.4 Result Aggregation

Our framework assumes that a parallel computation will end at a single node, which collects any pending outputs and combines them into a final result for the entire computation. Due to the broadcast mechanism used for all task outputs, however, it is possible that all nodes that receive a complete set of outputs can play this role. Such nodes broadcast the final result to the network, so that it can be received right away at the node that initiated the computation. In this way, our framework exploits the communication and computation diversity present in the distributed system.

3.3 The Rainbow Protocol

Next, we turn our attention to content distribution over a wireless ad hoc network. Properties specific to wireless multi-hop ad-hoc networks introduce complexity to the content distribution problem. The wireless medium is shared, rendering wire-line content distribution strategies that rely on multiple unicast flows inefficient. Similarly, the shared wireless medium also causes conventional flooding techniques (e.g., in [10]) to suffer from the "broadcast storm problem" [11], where relaying nodes experience heavy contention for the channel since their rebroadcast events are highly correlated in time. More advanced techniques that constrain flooding to a broadcast/multicast tree [12] require the building of a shortest path tree or a Steiner tree. However, physical phenomena such as fading, interference and capture effects [13] translate into transient changes in link quality that are difficult to predict, implying that the overhead of constantly rebuilding or re-ordering such trees can be prohibitively expensive.

Of particular note is the wireless-specific, hidden terminal problem [14]. In the classic case, hidden terminals cause collisions but, under the 802.11 carrier-sensing MAC,
hidden terminals may even prevent certain nodes from *transmitting* for prolonged periods of time. In scenarios where there are multiple clusters of nodes and content must pass through bottleneck, ``bridge" nodes, this can cause severe performance degradations. We term this phenomenon the *bridge lock-out problem*, and illustrate it using the simple scenario in Figure 4.

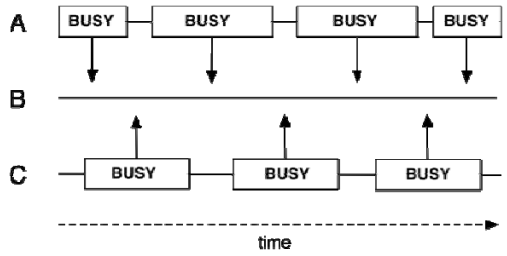


Figure 4: Illustration of the Bridge Lock-out Problem

We first observed the bridge lock-out problem on a real-world wireless testbed deployed in an office building and organized into a three-cluster topology. Simulating the phenomenon requires a detailed MAC model; we were able to reproduce it using the OPNET discrete event simulator in a scenario where clusters A and C had 5 nodes each, and B contained 1 node. With the 11 Mbps 802.11b modulation, and infinite offered load on all nodes, nodes in clusters A and C achieved a 1.8 Mbit/s average transmission rate, while node B achieved merely 7.9 Kbit/s.

Despite the challenges outlined above, wireless offers the opportunity to exploit the medium's broadcast properties. Due to the nature of wireless channels, a single transmission can be heard by multiple receivers---a property we term *broadcast advantage*. In comparison to multiple unicasts, broadcast advantage has the potential to significantly reduce the number of transmissions required for distributing a piece of content to multiple receivers, even under lossy conditions.

Our general approach to wireless content distribution is to use broadcasts to propagate content over multiple hops, as would be done in most flooding techniques. However, the shared nature of the medium raises the question of how to allocate channel share to nodes so that they can relay efficiently. To address this issue, we use a notion called *innovative content*} [15]. A piece of content is deemed to be "innovative" to a node if it represents a new piece of content which cannot be derived from existing content already held at the node. When network coding is used, innovative content will increase the
rank of the coded content the node holds. Clearly, transmission priority should be given to those nodes which are most capable of delivering innovative content to their neighbors. To determine which are "most capable", a node's content and link quality, relative to each neighbor, should be taken into account. We call a MAC that follows this policy a *content-directed MAC*. In contrast, we term protocols that do not use such policies as "content-blind".

Even with a content-directed MAC, efficient implementation of a wireless content distribution protocol can still be difficult, as all nodes must receive the content in its entirety. Flooding protocols are faced with the "coupon collector's problem" [16]. That is, if any node is delayed in receiving its last piece of content, the entire content distribution is delayed. More sophisticated relay schemes such as Selective Forwarding [17] are susceptible to high inefficiency: if only one node is missing a particular piece of the content, then
its broadcast retransmission is useless to all the other nodes that have already received it.

Network coding has been suggested as a way to overcome these limitations (e.g., [17], [18], [19], [15], [20]), as it has been shown to be an efficient reliable wireless multicast method which achieves a logarithmic reliability gain over ARQ mechanisms [21]. For this reason, and because our target environment is likely to have lossy wireless links, we employ network coding as a protocol building block.

A further benefit of network coding, particularly useful for the implementation of our content-directed MAC, is the convenience of using the rank information inherent to random linear coding [22] as a compact indicator of how much innovative content a node has, relative to its neighbors. Thus, rank serves as the lynchpin that ties together network coding (as an efficient and reliable wireless multicast method) with a content-directed MAC.

The Rainbow protocol is based on two key elements: (1) a mechanism to control access to the shared medium based on each node's content distribution performance, and (2) a network coding scheme for the outgoing data at each node. We detail these two elements in the following subsections.

3.3.1 Content-directed Medium Access Control

We base Rainbow's MAC on a two-step innovation reporting mechanism. First, on each outgoing data packet, a node piggybacks a short report containing the normalized rate at which it received innovative packets---packets carrying new pieces of content---from each neighbor over the past T_r seconds. The innovativeness of packets is determined by the content distribution layer, and so we describe the Rainbow MAC as content-directed. The normalized rate is computed relative to the sending rate, which is derived from sequence numbers; this way, the report implicitly includes the quality of wireless links. To control the size of this piggybacked report, we limit the number of innovation reports that can go into each packet to P. If a node has more than P neighbors to report on, then it will randomly select a subset of size P.

Second, using data from these reports, a node estimates the total *potential innovation* its transmission could provide to neighbors, by summing over all its neighbor nodes, and includes this total in future transmission reports. Nodes then select their transmission rates as follows. A node with the highest potential innovation transmits at the maximum link rate r_L, whereas other nodes send at some small, but non-zero *probing rate* r_P. This way, when the winning node runs out of innovative data to send, other nodes' potential contributions can be gauged from the data sent at the probing rate, and another node promoted to take over the channel.

Under this scheme, each node makes its transmission rate decision based on a comparison of potential contributions of its *one-hop* neighbors; however, the recipients of these contributions reside in the node's *two-hop* neighborhood. As a result, the scheme can properly schedule some difficult topologies such as the cluster-bridge topology from Figure 4; in that case, the bridge's upstream neighbors yield when the bridge receives all available content, and becomes the only useful transmitter among its one-hop neighbors.

3.3.2 Network Coding Layer

We adopt a random linear coding scheme for network coding, as detailed in [23]. The content to be distributed is assumed to consist of k content symbols $x_{i,i} \in \{1,...,k\}$, each of which is drawn from a base finite field K, typically GF(2^q), where q is a small integer such as 8. In coding and decoding, all arithmetic operations performed on symbols are in this base field K.

Under the random linear coding scheme, each transmitted symbol y_i is a random linear combination of l content symbols, i.e., $y = \sum_{j=1}^{l} c_{i,j} x_j$, where $c_{i,j}$, drawn randomly from the base field K, are coefficients of the encoding. A receiver receiving $L \geq l$ coded symbols $y_1,...,y_L$ will, with high probability, be able to recover the content symbols $x_1,...,x_l$ by solving the following system of linear equations:

$$\begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1l} \\ c_{21} & c_{22} & \cdots & c_{2l} \\ \vdots & \vdots & \ddots & \vdots \\ c_{L1} & c_{L2} & \cdots & c_{Ll} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_l \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_L \end{pmatrix} \quad (7)$$

To do so, the receivers need the coefficient vector $c_i = (c_{i,1} \cdots c_{i,l})^T$, so c_i needs to be transmitted along with the coded symbol y_i. In a packet-switched network, symbols are naturally grouped into packets. To reduce overhead, symbols transmitted in one packet share the same coefficient vector c_i. The overhead incurred by such a random linear coding scheme is thus $\eta = l/(m-l)$, where m is the maximal number of symbols that a packet can contain, given that most packets are of maximum size in typical content distribution applications.

To manage this overhead when distributing a large amount of content, the entire collection of content symbols is further broken down into *generations*, each consisting of l(m-l) symbols, or l packets, as suggested in [15]. The total number of generations will thus be $\lfloor k/(l(m-l)) \rfloor$.

We store the content in reduced row-echelon form. That is, upon the receipt of a coded packet, a node performs an (incremental) Gauss-Jordan elimination on the *coefficient matrix* $C = (c_1 \mathrm{L}\ c_L)^T$. When it terminates, it will report whether this packet has increased the rank of the matrix.

Whenever a node is allowed to transmit, it simply picks uniformly at random a generation among those coded generations of which it has received one or more packets. In theory, it should then form a random linear combination of all the packets it has in that generation. However, since we store the coefficient matrix in reduced echelon form, the node does not need to combine every single packet. Instead, it only picks those corresponding to the non-zero rows in the matrix since the span of these rows already contains all the information that can be derived from the entire collection of received packets in that generation.

The use of network coding is important for content-directed MAC, because it leads to small-sized reports that can piggyback on data packets---a single number, i.e., the rank of a node's coefficient matrix, suffices to describe the “innovativeness” of content held by that node. Furthermore, no additional control messages are needed, such as acknowledgments or descriptions of the content subset held by a particular node.

4.0 RESULTS AND DISCUSSION

4.1 SISR

In this section, we describe simulation and field experiments we performed to evaluate the effectiveness of SISR in localization with wireless nodes.

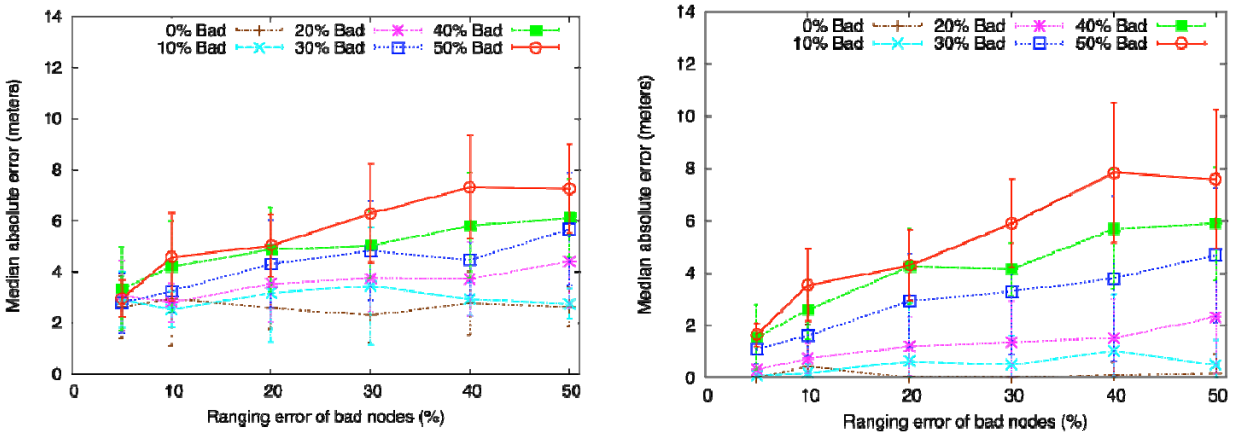
4.1.1 Simulations

In our first set of simulations, we want to determine how much SISR can improve upon MDS-MAP(C,R) in non-convex, "C"-shaped topologies, under various error conditions. We first outline the experimental procedure and follow with simulation details. For a given topology and set of error conditions, we compute the MDS-MAP(C,R) localization solution. This is then used as an initial approximation for SISR localization. We then compare the accuracy of the resulting solutions from the two methods.

For each trial, we generate a topology where 147 nodes are placed uniformly at random within a "C"-shaped region in a bounding area of 100m x 100m. An example of such a topology is shown in Figure 2(a). Each node has a simulated radio range of 16.5m, which results in a multi-hop topology. Each trial is also described by two experiment conditions governing error. The first experiment condition is the percentage of "bad nodes" within the population (0%, 10%, 20%, 30%, 40%, 50%). A node is assigned to either a "good" population, in which case it has no error in ranging, or a "bad" population, in which case it experiences ranging error $\varepsilon \sim N(\mu_{bad}, 0.2\mu_{bad})$. The second experiment condition is μ_{bad} (5%, 10%, 20%, 30%, 40%, 50%). Note that a "bad node" has ranging error only from itself to other nodes and not vice versa; this results in asymmetric range measurements, a more realistic scenario.

We have 36 combinations across the two experiment conditions, for a total of 36 simulations. Each simulation consisted of 20 trials on distinct topologies. The average median absolute localization errors are shown Figure 5. First, consider the results in Figure 5(a), where MDS-MAP(C,R) is shown to be relatively insensitive to all values of ranging error, when the percentage of "bad nodes" is low (<30%), but never gives particularly accurate results. All MDS-MAP(C,R) solutions had at least 2m error, even when there are no "bad nodes" in the population.

In contrast to MDS-MAP(C,R), SISR is capable of providing much more accurate results---less than 2m error---when the percentage of "bad nodes" is less than 30%, as shown in Figure 5(b). For these conditions, due to (1) the "snap-in" behavior of SISR, (2) the fact that "good nodes" have no measurement error, and (3) a good initial approximation afforded by MDS-MAP(C,R), the resulting localization solution is not significantly affected by the magnitude of the ranging error. Of course, in a real setting, we cannot expect "good nodes" to be absolutely correct in their range estimates; we will explore one such real setting in Section 4.1.2, where we describe our field experiments.



(a) (b)

Figure 5: A Comparison of the Accuracy of (a) MDS-MAP(C,R) vs. (b) SISR

4.1.2 Field Experiments

In this section we describe a preliminary localization experiment using actual 802.11 radios in an outdoor environment. The goal of this experiment was to validate SISR localization using real systems and radios on real-world data and especially in the "bad node" scenario. We first describe our experimental setup and explain how we collected and processed the RSSI measurements. We then compare the localization results returned by MDS-MAP(C,R), the Lorentzian estimator (a well-known robust estimator), and SISR. In comparing the Lorentzian with SISR, we tuned the former such that it closely matched the shape of the SISR function at $\tau = 0.25$m. As we will see, SISR effectively mitigates the effect of the bad node. However, since the topology in this experiment is convex and nodes reside in a single broadcast domain, we do not expect to observe significant performance gains over MDS-MAP(C,R). Nevertheless, we do want to verify that our SISR implementation is correct in the sense that it should perform comparably to MDS-MAP(C,R). As future work, we will conduct field experiments with non-convex topologies to demonstrate SISR's performance gain, as predicted by our simulation results in Section 4.1.1.

To convert RSSI into range measurements, we have developed a path-loss model, and validated the model via field measurements. We chose a flat grass field near a football stadium as the experiment site. There were no major buildings or other obstacles nearby, and only faint ambient beaconing traffic from 802.11 base stations. The nodes we used were the One Laptop Per Child (OLPC) computers, which contain an x86 compatible AMD processor and run Linux. The radio in the OLPCs is a Marvell 8388 802.11 b/g internal USB module. We placed the radios into ad-hoc mode, and took steps to disable 802.11 beaconing and IBSS formation which could cause unexpected network splitting or otherwise interfere with experiments [24]. Lastly, we modified the Linux kernel to provide Received Signal Strength Indicator (RSSI) values as metadata alongside any received UDP packets in order to facilitate link measurements.

We structured our experiments as a series of basic one-way signal strength measurement operations between pairs of nodes, performed as follows. One node of the pair would transmit for a relatively long period of time, broadcasting a stream of 64-byte UDP probe packets at 2Mbps modulation and on 802.11 channel 4. The receiver node would listen for these probe packets, logging each packet's transmitter address and RSSI. Note that it is convenient to perform these basic measurement operations in parallel by having multiple receivers capture a single transmitter's probes.

Due to environmental effects, RSSI measurements may be subject to fading fluctuations. At the same time, the RSSI reporting mechanism inside our radios' firmware is an undocumented black box, and may report values which deviate from the actual received power. Therefore, in order to obtain a more reliable RSS estimate, we use a simple windowing heuristic to filter the RSSI data. For node i receiving from node j, we take n total RSSI samples, divide these into w consecutive windows, take the maximum RSSI value in the window and then take the median of these values to be the "representative" RSSI of transmissions from node j. Of course, the more samples taken, the more accurate the representative RSSI. From our field data, we have found that the RSSI heuristic converges in as few as 1000 packets (~300ms transmission time), suggesting that the wireless ranging approach we have adopted can be done quickly. We arranged 20 testbed nodes within a 25m x 25m square region, which we found to be the approximate area of a broadcast domain for nodes that are placed on the ground (the diagonal, at ~36m, is the limit of transmission range).

Among the nodes in our testbed, a single node X proved to have an interesting failure mode. Specifically, this node functioned normally when transmitting, but would consistently report abnormally low RSSI values for any received packets (e.g., ~30dB lower than other nodes for a transmitter only 1m away). Likely due to faulty hardware, this behavior is ideal as a test case for fault-tolerant localization methods such as SISR or MDS-MAP.

Figure 6 shows the localization solution given by SISR (red diamonds) and MDS-MAP(C,R) (black diamonds). The ground truth locations are indicated by grey circles and the lines connecting the localized positions indicate node identity. Figure 6 visually verifies that most nodes are accurately localized by SISR, with the obvious exception of one node, which has been localized to an impossibly distant position. Naturally, this is node X, whose ranging measurements were faulty across the board. This is the hallmark behavior of the SISR method – relatively less effort is made to minimize the error due to bad measurements at the expense of the good ones.

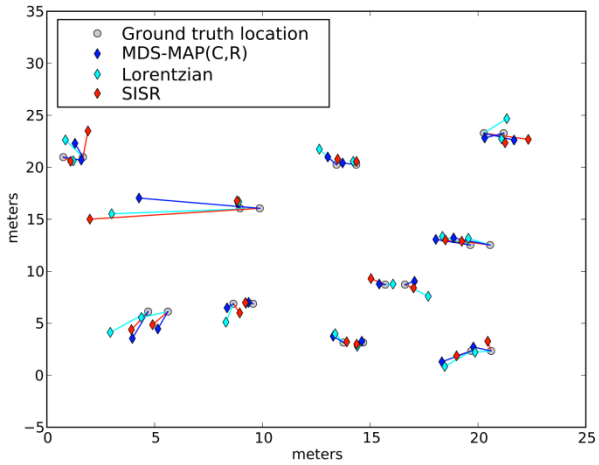


Figure 6: SISR (red), Lorentzian (Cyan), MDS-MAP(C,R) (Blue) Localization Solutions

SISR gives a median absolute localization error of 0.90m and MDS-MAP(C,R) gives 0.89m; in contrast, the Lorentzian estimator gives 1.27m. The CDF of absolute localization error across nodes is shown in Figure 7. The SISR curve (red) tracks very closely to that of MDS-MAP(C,R) (blue). The two methods performed comparably, due to the relatively uniform, convex topology and lack of potentially distorting multihop links. As a comparison, the Lorentzian tracks under the MDS-MAP(C,R) curve, indicating that it performs worst overall. This shows that not all robust estimators solve the localization problem equally well and that SISR is particularly suited for the task.

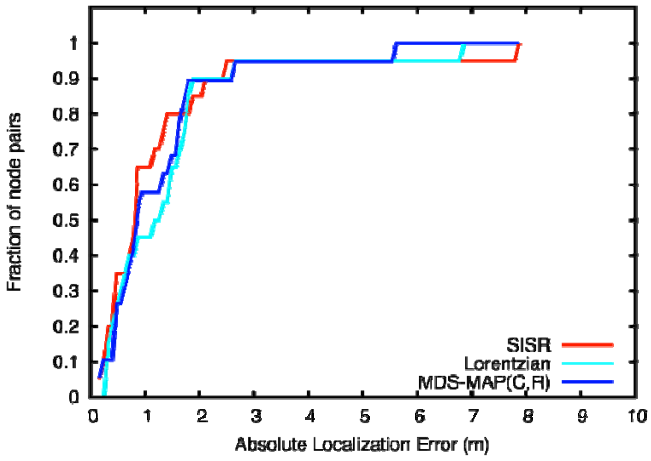


Figure 7: CDFs of the Absolute Localization Error across All Nodes as Given by SISR (Red), the Lorentzian Estimator (Cyan) and MDS-MAP(C,R) (Blue)

4.2 DWARF

In this section, we present our evaluation of DWARF using distributed speaker identification as a benchmark application. Speaker Identification (SID) is the task of determining a person's identity from her voice sample, independent of the words or language she speaks. Typically, SID systems enroll a set of speakers of interest (e.g., authorized personnel). An *open-set* SID system is capable of identifying a non-enrolled speaker as "out-of-set", rather than simply reporting the best "in-set" match. Such systems have military applications including fratricide reduction, enhanced force protection by blue force tracking, or surveillance target identification.

Our open-set SID system is a Gaussian Mixture Model (GMM) classifier based on [25]. Models are trained from Mel-cepstra acoustic features, which capture snapshots of dominant vocal resonances. We divided the NIST TIMIT corpus of 630 speakers, into two sets: 530 in-set speakers and 100 out-of-set. Once 530 speaker models were trained, we scored the models against a separate group of in-set and out-of-set speaker utterances and adjusted a decision threshold value to equalize the number of false acceptances and false rejections. In the recognition phase, feature vectors are extracted from an unknown speaker's utterance and scored against each speaker model. If the maximum normalized score is greater than the decision threshold, then the unknown speaker is declared to be in-set and her identity is the speaker whose model produced that maximum score. Otherwise, the unknown speaker is declared to be out-of-set.

This recognition computation grows linearly with the number of enrolled speakers and is computationally intensive---with 530 speaker models, computing the result for one 5-second audio clip on a single 1GHz node in our wireless testbed takes 11 seconds on average. Our goal here is to significantly speed up SID, and we do so via parallel processing. A simple and natural decomposition of this problem is to divide the database of speaker models across a set of computation nodes. For example, if there are 10 nodes, we divide the 530 speaker models into 10 disjoint subsets, each with 53 speaker models. When presented with a speaker utterance, each node is responsible for scoring it against a different speaker model subset in parallel. Subsequently, each node reliably broadcasts its calculated scores to all other nodes. Once a node has received all 530 scores, it can perform adjudication locally.

Consider a set of 16 microphone sensors arbitrarily placed in a square area of 10-ft unit length. Each sensor is connected to a computation node that is equipped with a 802.11b/g Wi-Fi network interface, modest computing resources and is a participant in DWARF. At an arbitrary location within this area is a single, stationary speaker (i.e., no co-channel speech present), whose sound pressure level is 30dB at the source. At a second arbitrary location is a stationary audio noise source, emitting pink noise (10dB at the source). Nodes do not know their own locations, nor those of the speaker or noise source. For our experiments, we have chosen speaker, noise source and sensor locations as shown in Figure 8.

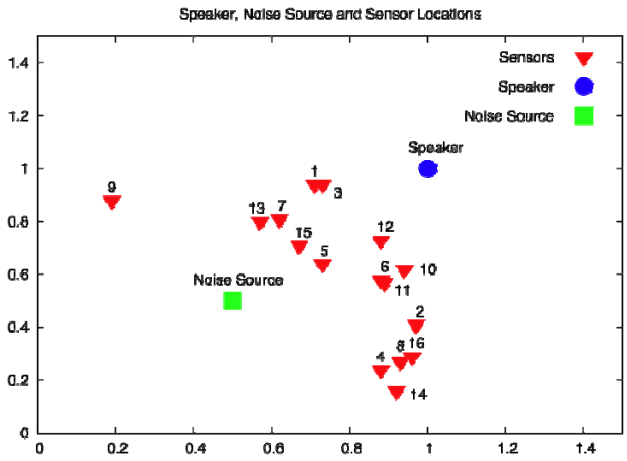


Figure 8: Speaker, Noise Source and Sensor Locations in the Experiment Scenario

Each node is responsible for monitoring its acoustic neighborhood for speech. We assume that each node digitizes and segments its audio stream into homogeneous speaker segments but due to the arbitrary locations of the speaker and noise source, each node may capture such segments with varying signal-to-noise ratios (SNRs). The open-set SID algorithm will perform poorly when presented with a speech segment that has low SNR. However, with multiple nodes distributed at various geographical locations, we are able to exploit sensor diversity by first determining which node has the highest SNR and using only that node's speech segment as input for all the other nodes.

Specifically, our distributed SID method adheres to the following task pipeline:

1. Each node calculates the SNR of the first 250ms of the speech segment generated by its own microphone.
2. All nodes participate in a distributed SNR election that determines the node with the highest SNR.
3. The election winner calculates a speaker feature matrix from its input speech segment and reliably broadcasts this matrix to all other nodes.
4. Upon receipt of the feature matrix, each node scores the input against its subset of speaker models in parallel. When complete, the resulting score vector is reliably broadcast to all other nodes.
5. When a node has aggregated all the score vectors, it performs adjudication on the scores and returns the SID result.

We deployed DWARF and our distributed SID application on a wireless ad-hoc network of 32 nodes. Each node is a 1GHz VIA C7 processor with 1GB RAM, a 1GB flash drive, and a USB 802.11b/g wireless network interface (VIA chipset), running Ubuntu Linux 7.04. We have modified the VIA Wi-Fi driver such that IBSS beaconing is disabled and variable rate broadcast is enabled. For all our experiments, we use 11Mbps modulation for broadcast packets.

For practical reasons, we simulated the noisy speech segment heard at each of the 16 sensor locations by adding pink noise to a noise-free sample. The amount of noise added reflects the expected SNR at each sensor location. Thus, the input into the distributed SID application running on each DWARF node is a WAV file containing a simulated noisy speech segment.

The computation nodes we use in our experiments were not geographically distributed as shown in Figure 8, but were instead rack-mounted as shown in Figure 9. We have previously verified that the configuration of nodes on the rack produces link qualities that are very similar to nodes placed at random within a 10ft x 10ft area. While simplified, this configuration still employs a true wireless environment in which to measure DWARF performance. In contrast, had we instead emulated wireless channel characteristics over wired Ethernet, it would have been difficult to achieve the timing accuracy required to accurately model collisions.

Figure 9: Photo of Wireless Ad-hoc Network Consists of 32 Computation Nodes, Each Running DWARF

Note that while the experiment scenario specifies 16 simulated sensors, our testbed consists of 32 computation nodes. This gave us the opportunity to increase the level of parallelism during computation by simply connecting up to two computation nodes to each sensor. To stress our system, we required a larger speaker model set than the TIMIT corpus could provide. We artificially inflated the size of the speaker model set by duplicating each model, thus requiring the distributed SID application to compute scores for 1060 speaker models (530 unique).

Using the experiment scenario and setup described above, we evaluated our system according to three major criteria: (1) speed-up due to parallelization, especially under faulty conditions; (2) the amount of system overhead resulting from DWARF; and (3) the speed-up due to mobile nodes rejoining a computation after a period of departure.

4.2.1 Speed-up Due To Parallelization

Since one major goal in our work is to reduce the execution time of speaker identification, we are interested in measuring the speed-up due to parallel processing on DWARF. To do so, we measured the average running time (20 trials) of the distributed SID application, as shown in Figure 10. Note that since all nodes listen for a complete set of score vectors before performing local score adjudication, transient fades in the wireless links can result in different adjudication start times. This, in turn, leads to different completion times. As a result, we define “running time” as the time elapsed until the first adjudication result is reported by any node.

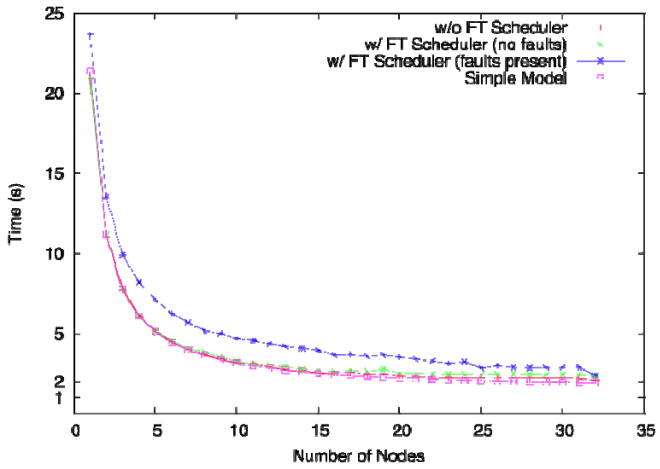


Figure 10: A Comparison of Average Running Time of Distributed Speaker Identification over DWARF, Under Different Conditions

We compare three experiment conditions to a simple simulation model (purple line) in which we take into account our implementation overhead (audio input pre-processing, timeouts) and data transmission time under 10% packet loss. The simple model represents the optimal speed-up that can be achieved with our implementation in the absence of node failures (i.e., no faults present).

Our baseline measurement in the experiments is the running time of the distributed SID application over DWARF without the fault-tolerant scheduler in the loop or any faults. To show speed-up, we vary the number of nodes in the computation group from one to 32 (red line). Next, we establish that the fault-tolerant scheduler does not introduce significant overhead. To do so, we performed the same experiment, but with the fault-tolerant scheduler handling the invocation of the computation tasks (green line).

Finally, we measured system performance under faulty conditions (blue line). Starting with 32 nodes in the computation group, we inject synthetic faults by terminating the fault-tolerant scheduler on k randomly selected nodes immediately after it has broadcast its first scheduled task. For the blue line in Figure 10, the x-axis represents the number of nodes that remain alive after fault injection (i.e., n=32-k). Since the system initially starts with 32 nodes, the computation is decomposed into 32 tasks; as faults occur, the scheduler detects which tasks were interrupted and reassigns these to run on the nodes that remain in service. In this way, we demonstrate that DWARF successfully recovers and drives the computation to completion. However, because there are fewer working nodes, we expect the average run-times to increase with the number of faults and indeed, we observe this behavior.

4.2.2 Overhead in the DWARF System

In all three cases, the results in Figure 10 show an approximately ten-fold speed-up when using 32 nodes, as compared to using a single node. While we show that our implementation matches well with our simple model, the speed-up grows only marginally beyond ~20 nodes and is short of being linear. Investigating further, we decomposed our running times into their two major constituent parts---the SID computation (green line) and DWARF overhead (blue line)---as shown in Figure 11. Note that the DWARF overhead (blue line) eventually begins to dominate the SID computation time (green line), even though we still observe a marginal speed-up as the number of nodes increases. This is because the rate of change of overhead is lower than that of the computation time of a task.

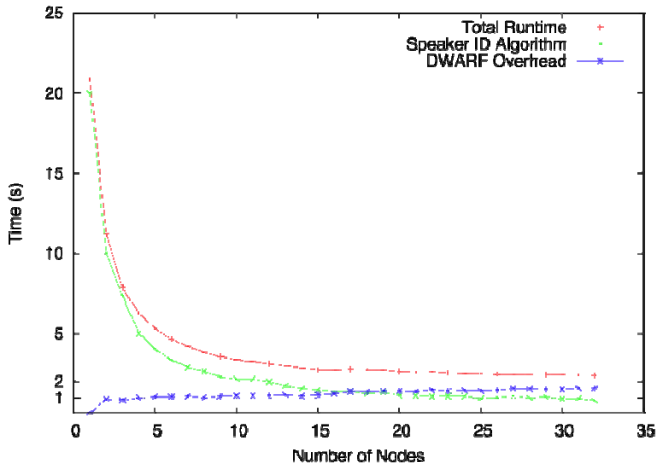


Figure 11: Average Running Time of Distributed SID over DWARF with a Varying Number of Nodes and the FT Scheduler Enabled, But with No Faults Present

In general, the cost of reliably broadcasting data to every node over lossy wireless links is the time the sender requires to detect a lost packet at the receiver(s) and to retransmit. This cost rises with the number of nodes because the probability that any one node loses a particular packet increases. Retransmission efficiency decreases in tandem, further increasing overhead---e.g., if only 1 out of 32 nodes loses a particular packet, then its retransmission is useless to the other 31. Meanwhile, with an increasing number of nodes, the task size per node decreases, since we decompose the overall computation into smaller units. The overall effect is a decrease in the computation-to-I/O ratio, implying a reduction in parallel speed-up. One possible way to alleviate this general problem is to relax the condition of reliable broadcast. For instance, if 90% of participating nodes completely receive the data being broadcast and immediately start computation, they may return a result more quickly than if they wait for the remaining 10% to finish receiving. We will explore this trade-off in the future.

Figure 12 further decomposes DWARF overhead into its constituent parts, allowing us to identify the specific I/O bottlenecks. It is important to note that the election overhead (red line) remains low as the number of nodes increases, meaning that the cost of exploiting sensor diversity is minimal.

In contrast, the speaker feature matrix broadcast time (green line) experiences a 1.4-fold increase from 2 to 32 nodes; result aggregation time (purple line) shows a 2.9-fold increase over the same range. While both are partially due to the increasing number of nodes contending for the channel, variation in computation task completion time (e.g., with 32 nodes, μ=0.686s, σ=0.029s) also contribute to the increase in aggregation time. By definition, aggregation cannot complete until the last remaining task finishes. With 32 nodes, we have observed a mean difference of 0.112s (σ=0.040s) between the first and last task completing, suggesting that the actual amount of time spent communicating results is smaller and that much of the aggregation overhead can be attributed to waiting for the last result to become available.

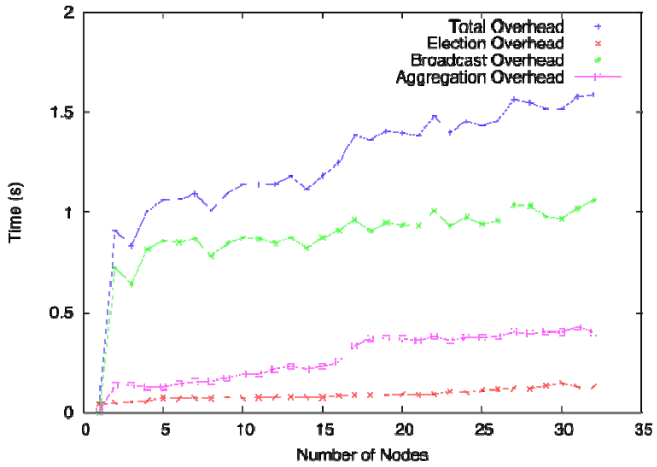


Figure 12: A Breakdown of Average Overhead Incurred by Distributed Speaker Identification over DWARF

4.2.3 Mobile Nodes

Finally, we show that DWARF can support node mobility. Consider a scenario where an unmanned aerial vehicle (UAV) loaded with computation nodes participates in a distributed computation with nodes on the ground. The UAV may become connected or disconnected to the ground nodes, depending on its current location. We demonstrate that DWARF can support and take advantage of this depart-and-return mobility model. In this experiment, we employ two indoor nodes running distributed SID on DWARF, as illustrated in Figure 13. The computation is divided into 32 tasks and, initially, the two nodes are within range when the computation starts. Subsequently, one node moves along the tour circuit shown, departing from communications range and then returning. We vary the duration of the mobile node's tour and show that, upon return, the node is automatically re-incorporated into the distributed computation by DWARF. Note that in this single experiment, we address two distinct mobile scenarios: node exit and node entry. The latter case demonstrates DWARF's ability to add mobile computing resources on-the-fly.

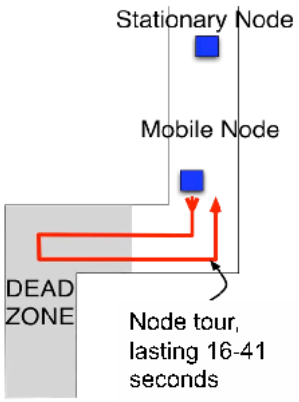


Figure 13: A Diagram of a Simple Mobility Experiment

Figure 14 shows the average running time (3 trials) of our SID application across different tour durations. First, we present two control measurements which supply the bounds to the running time: with a single node, the job takes 23.7 seconds to complete (upper bound; pink bar); with two nodes, it takes 13.5 seconds to complete (lower bound; green bar). The blue bars represent the running times across various tour durations, and show that running time decreases with faster tours (i.e., earlier return of the mobile node). This demonstrates that, upon return, the mobile node was able to help speed the computation. Note that even for the longest tour (41s), where the mobile node returned after the stationary node had finished the entire computation, we still experience speed-up. This is because the mobile node remains in range at the beginning of the tour and is able to complete some tasks and report their results before moving out of range.

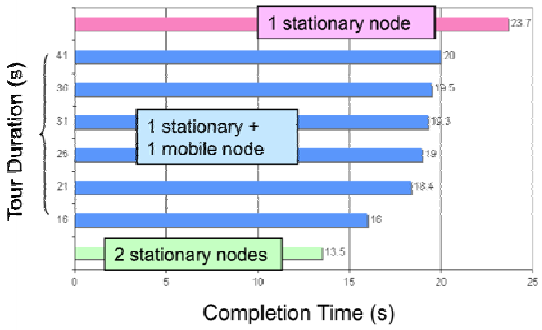


Figure 14: Average Running Time Decreases with Faster Tours of the Mobile Node

4.3 Rainbow

We performed outdoor field experiments on a large, flat athletic field on the Harvard campus in Cambridge, MA. Our testbed consisted of 29 OLPC XO Beta-2 nodes. These are i386 compatible systems based on the AMD Geode GX processor running at 366MHz, and equipped with 128MB RAM. Each system has one Marvell Libertas 88W8388 802.11b/g radio, with tunable transmit power. All of the nodes were placed on the ground. We used broadcast Ethernet packets at the 2Mbit/s rate for all protocol implementations.

We focused our experimental work on a class of topologies where nodes reside in clusters. Such topologies are interesting because they require that protocols cope with the high density of intra-cluster nodes in single collision domains, as well as inter-cluster multi-hop communication. We ran the majority of experiments on a 6-cluster topology shown in Figure 15; other experiments were run only on clusters 1-3. Nodes in the individual clusters are spaced closely---at most 12 ft apart---so they have almost lossless links to each other. Regardless of the topology, the content source was always a single node residing in cluster 1.

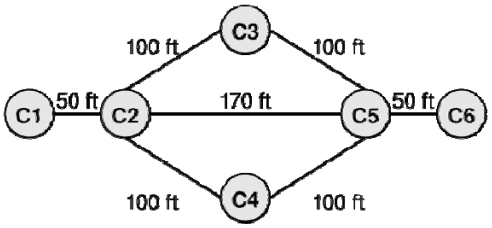


Figure 15: Six Cluster Topology Used in Rainbow Experiments

We compare the performance of three flooding protocols for content distribution: Blind (i.e., content-blind), Uncoded, and Coded. The Coded protocol is the Rainbow protocol as described above. We use the other two protocols as points of comparison, and present their design here for completeness.

Blind flooding is a classic flooding protocol enhanced by using a smaller-than-unity relaying probability. Such probabilistic flooding has been proposed in the literature to address the redundancy of broadcast transmissions in the wireless medium [11]. Classic flooding is content-blind, meaning transmission priority is not moderated. In order to use the protocol to send large files the source node splits a file into packet-sized chunks, and floods the chunks sequentially. Once the last chunk is flooded, the source repeats the sequence, starting over from the first chunk.

We have also designed and implemented a content-directed uncoded flooding protocol. The basic design is similar to the Selective Forwarding strategy [17], with optimizations specifically designed for wireless networks, and works as follows. The content to be distributed is broken into x fixed-size chunks, each of which consists of y symbols. Every T_b seconds, nodes broadcast an x-bit vector, in which the bits corresponding to the chunks they have are set. By broadcasting its content bit vector, a node implicitly requests missing content from its neighbors. With the knowledge of what neighbors have, a node sorts its chunks according to the rarity in the neighborhood, and then transmits starting from the rarest one. One of our optimizations is that nodes yield the medium in the presence of nodes with more content. That is, if a node detects another in its neighborhood with more content than itself, it will transmit at a lower probe rate r_P. Only the node that has the most content in its neighborhood will transmit at full link rate r_L. We refer to this protocol as “Uncoded”.

It is important to note that the Uncoded protocol is not simply the Coded protocol without network coding; they have distinct MAC mechanisms chosen to fit the situation where rank information is absent. Recall that the Coded MAC in Rainbow uses historical performance (i.e., innovation reports) over a two-hop neighborhood in deciding whether a node should transmit at the maximum link rate r_L. Network coding enables us to use rank information as a natural way of expressing a transmitter's past performance. Moreover, the stability of a rank-based innovation metric is desirable; good historical performance generally indicates good future performance under network coding. While a similar mechanism in the Uncoded case is also possible in principle, the analogous innovation metric is less stable. For example, under the ``rarest first'' policy, the node with the best historical performance may abruptly run out of innovative chunks to send; this can lead to thrashing when deciding who should transmit at r_L. To address this problem, our Uncoded MAC makes this decision based simply on the amount of content each node in a one-hop neighborhood possesses. While the Coded and Uncoded MACs are not exactly analogous, the Uncoded protocol is a reasonable content distribution scheme and represents a fair point of comparison.

In the implementation of these three protocols, the neighbor information is timed out if a node has not received any updates from that neighbor for an interval of T_o seconds. The key metric we use to evaluate performance of the three protocols is the completion time, i.e., the time required for all nodes to obtain the file being distributed. Such a performance metric has also been used by, e.g., [22]. Note that this metric is sensitive to the presence of any poorly performing nodes; for this reason, we limited the experiment running times to a reasonable upper bound that let most nodes complete, yet kept the experiments practical. We have also included

several empirical probability distribution functions in the next sections to demonstrate that the completion time can serve as a representative indication of how good a contention distribution protocol is. In particular, we used a small multiple of the content transmission time at link speed. The size of the file we distributed was 6.1MBytes, which at the 1.7Mbit/s link rate of our testbed takes about 30 seconds to transfer. We limited the experiment run time to 300 seconds.

4.3.1 Comparison of Blind, Uncoded and Coded Flooding

We report the completion times of the three flooding protocols for content distribution: Blind, Uncoded, and Coded (i.e., Rainbow). The individual node completion times are plotted as empirical distribution functions (EDFs).

Figure 16 shows the performance of five different Blind flooding instances with relay probabilities p=0, 25%, …, 100%, as compared to that of Coded flooding (Rainbow). First, note that 0%-flooding completes an initial set of nodes earlier than any other protocol. This behavior shows that simple single-source broadcast can be effective in low packet loss environments with nodes in a single collision domain. However, because relaying is not permitted, nodes out of reach of the source cannot obtain the content. When the relay probability is above zero, we can see that while more nodes get portions of the content, the completion times of the nodes nearest to the source increases. The reason is that channel share is allocated to more nodes than just the source, slowing down the useful transmissions.

Figure 16 also showcases the adaptability of the Coded protocol (Rainbow). That is, Rainbow performs well both in the single collision domain, where its performance is within an order of magnitude of the 0%-flooding, and in the whole topology, where it is the only protocol to complete before the 300s deadline.

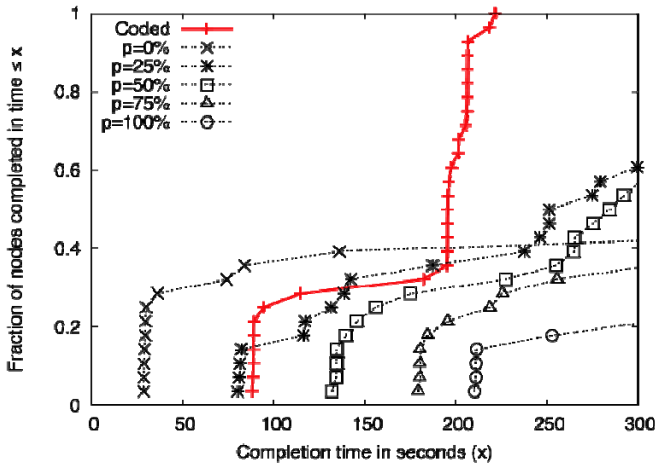


Figure 16: Empirical Distribution Function of the Completion Times for Blind and Coded Flooding

Next, we compare the performance of Coded and Uncoded flooding, while at the same time showing that the completion times are stable across different runs. We present this comparison in Figure 17, where we plot the completion distribution functions of three runs each for both Coded and Uncoded. The results show that Coded completes earlier than the Uncoded in all cases. Specifically, when all the nodes in Coded complete, no more than about 60% of nodes in Uncoded have completed.

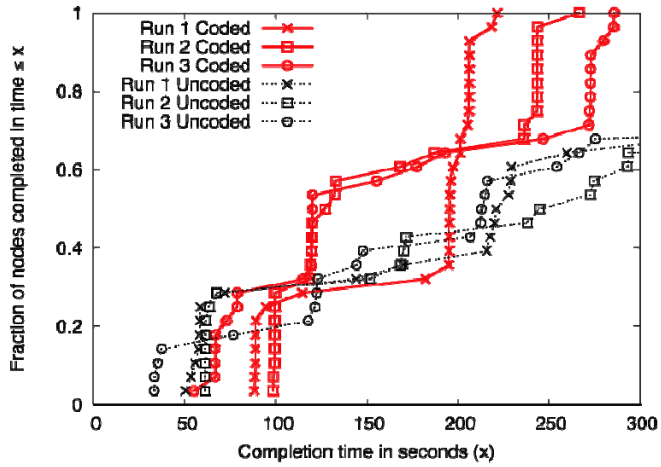


Figure 17: Three Runs Each of Coded and Uncoded Flooding

4.3.2 Effect of Cluster Size

We ran the Coded and Uncoded protocols on clusters 1-3 of Figure 15, while varying the size of those clusters from 1 to 5 nodes. We present the results in Figure 18. The key observation we make is that Uncoded performance is significantly degraded with the largest two cluster sizes, 4 and 5. In contrast, the Coded performance remains relatively stable. The key reason for the performance degradation is likely congestion, because unlike Coded, Uncoded allows multiple nodes to transmit at r_L. We provide some qualitative evidence of this in the next subsection.

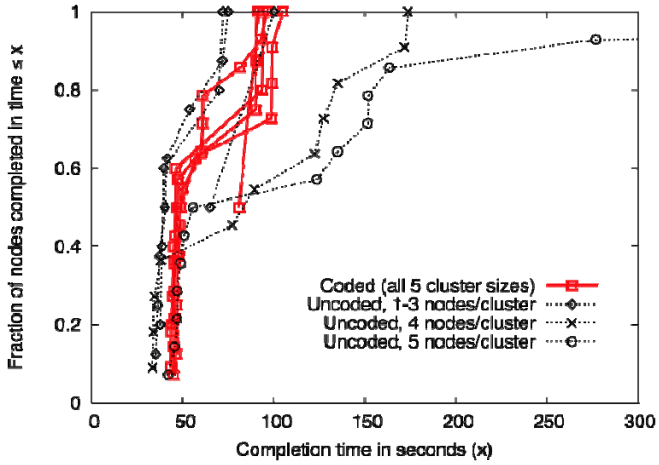


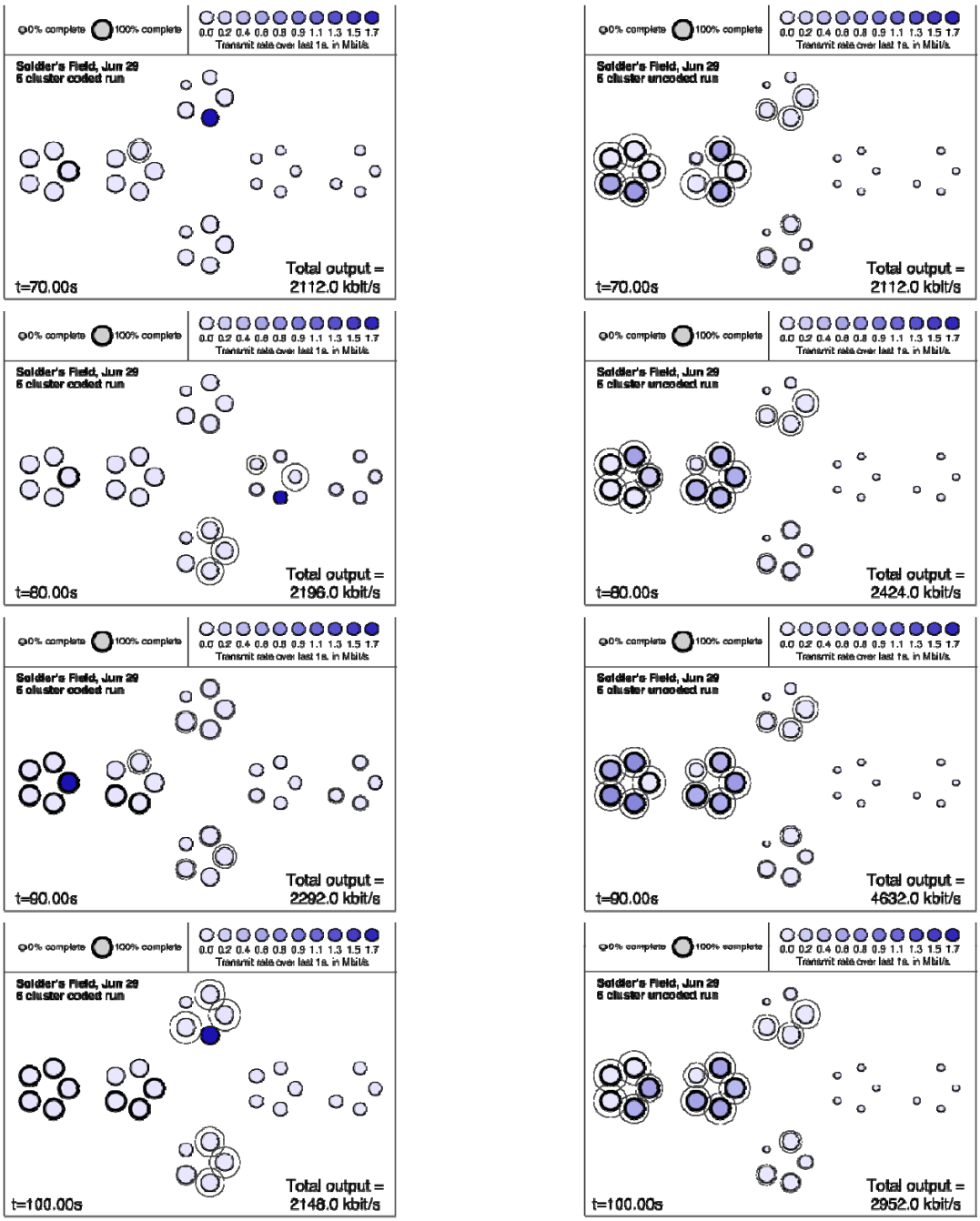
Figure 18: Performance Results of Coded and Uncoded Runs when Varying Cluster Sizes

4.3.3 Dynamic Behavior of Flooding Protocols as Observed by Visualization Tools

We created a set of visualization tools to aid protocol development by inspecting the dynamic behavior of the system. The tools allow us to play back traces collected during experiments and animate several time-varying metrics on a single plot. As an example of how we used the visualizations to gain insight into protocol performance, Figure 19 shows snapshots of Coded and Uncoded runs in the middle of a content distribution, where the content file propagates from the leftmost cluster to the rightmost. In the figures, a node's circle grows as it receives an increasing percentage of the file. The size of a ring around a node's solid circle represents the magnitude of redundant packets received in the past second, where redundant packets are those containing pieces of content already present at a node. When a node transmits at a higher rate, its solid circle has a darker color.

The key metric provided by the snapshots is the transmission rate of the individual nodes at a given time. As we can see, with the Coded protocol only one node transmits at a time; furthermore, the transmitting nodes are activated roughly in a round-robin order, each time delivering a small chunk of the content. In contrast, under the Uncoded protocol multiple nodes transmit at once, leading to collisions at the desired recipients and a slowdown in completion time.

The Coded snapshots provide insight into how Rainbow avoids the bridge lock-out problem. Consider the bottom-most Coded snapshot (left panel) corresponding to 100s from start of experiment. The only transmitting node at that time is in cluster 3, acting as a bridge between clusters 1, 2 and 5, 6. This node is able to transmit because none of its neighbors have been able to deliver innovative packets at a better total rate; therefore, per our MAC scheme, the other nodes yield.



Coded Uncoded

Figure 19: Snapshots of Coded and Uncoded Protocols at Time Points 70-100s, 10 Seconds Apart

5.0 CONCLUSIONS

5.1 SISR

Via simulation and field experiments, we have validated that our SISR localization method can tolerate errors in ranging measurements by de-emphasizing bad data while accentuating more accurate readings. In particular, SISR can accurately localize difficult non-convex topologies, even in the presence of non-Gaussian errors in ranging. Moreover, SISR is tunable, in that it can be made effective for different environments with different error conditions. We note that SISR is a general method and can be applied towards any ranging technique that experiences gross error, e.g., acoustic time-difference-of-arrival (TDOA). In the future, we plan to extend SISR to distributed and cluster-aware localization, where local, reliable clusters would first be localized, and later stitched together to form a global topology by using SISR on inter-cluster ranging measurements. SISR also suggests a natural way of pruning bad nodes from the network, which we plan to implement: by comparing the unshaped residual (conventional least squares) residual of nodes, outliers can be rejected. Such node discrimination may be critical to the performance of higher-level network applications.

5.2 DWARF

We demonstrated the capabilities of our Distributed Wireless Application Runtime Framework (DWARF) via an example distributed speaker identification application. Our major contributions are as follows.

First, we have shown that such applications, when deployed on DWARF, can experience significant speed-up due to parallelization. In general, applications that require data broadcasts are particularly well-suited to our framework, since DWARF's wireless backplane utilizes the medium's broadcast advantage to reduce I/O overhead in data distribution. Further, we have demonstrated that 802.11b/g at 11Mbps is more than sufficient to support compute-bound applications such as the distributed speaker identification problem considered in this paper. Additionally, application porting or development on DWARF is simple for application programmers, requiring only task binaries and a task dependency graph be supplied.

Second, DWARF-based applications automatically gain the fault-tolerant capabilities inherent to the framework. DWARF's fault-tolerant task scheduler enables computations to complete in spite of node failures (e.g., resulting from battery depletion in mobile nodes or traveling out of radio range) while making efficient use of the available parallelism to decrease running time. DWARF recovers from transient failures of wireless links resulting from fading and radio interference by automatically detecting task interruption. Moreover, DWARF can reassign tasks to other nodes upon node departure and re-tasking nodes upon return. The fault-tolerant scheduler accomplishes all these while honoring task dependency conditions and giving priority to critical tasks whose completion will allow a relatively larger number of tasks to be run in parallel in the future.

Third, and most importantly, we demonstrate two important interactions between our architecture and the algorithms it can support. First, the distributed nature of DWARF naturally lends itself to algorithms where sensor diversity can be exploited. In our distributed speaker identification implementation on DWARF, the SNR election plays a critical role in calculating accurate results. Without leveraging this advantage of sensor diversity, even the most cleverly-designed signal processing algorithm would be ineffective if sensor data with very low SNR were used as input data. DWARF is architected such that applications can easily capitalize on such diversity. Second, sensor diversity coupled with local wireless ad-hoc networking enables nodes to share sensor data and to quickly make local decisions without the need for communicating with a centralized arbiter over a slower, possibly unreliable or unavailable, long-haul connection (e.g., satellite). This is particularly advantageous to applications at the tactical edge, where computation results must be quickly produced and are typically consumed locally. Where data transmission over a long-haul link is necessary, DWARF can be used to pre-process and reduce the data, making more effective use of such narrow, shared uplinks.

5.3 Rainbow

We have designed and implemented Rainbow, a protocol for content distribution over multi-hop wireless ad-hoc networks. Rainbow was implemented and validated on a testbed of 29 XO Beta-2 development laptops from One Laptop Per Child. We compared Rainbow to two other protocols: content-blind probabilistic flooding and content-directed uncoded flooding. We have found that Rainbow outperforms probabilistic flooding by 1.3-fold and content-directed uncoded flooding by 1.9-fold, as calculated by the time required for 50% of nodes to reach completion, in the best cases. The reasons for these gains are outlined below.

By virtue of being content-directed, Rainbow is able to optimize medium access control for content delivery, previously impossible under traditional content-blind protocols. The key principle we utilize is that relaying nodes should only occupy the channel under a combination of two conditions: (1) when they have good links to their neighbors and (2) when the innovative content they possess can help the largest number of their neighbors. As a result, under Rainbow, only the nodes that are most capable of delivering innovative content at a given time gain access to the channel. This minimizes exposure to the broadcast storm and bridge lockout problems, and thus makes Rainbow well-suited for multi-hop topologies with bottleneck links, where the lockout problem has greatest impact.

We take advantage of network coding to efficiently implement Rainbow's content-directed MAC. Innovation reports, i.e., those based on rank information, are small enough that they can be piggybacked on coded data packets, allowing nodes running Rainbow to react quickly to transient changes in link quality and content quality of nodes. No separate protocol messages are needed to communicate completion status. Network coding does incur computation overhead at nodes by performing rank calculation and encoding/decoding, as well as communication overhead in transporting coefficients of random linear combinations. The results of this paper suggest that performance gain resulting from using network coding outweighs its overhead in our content-directed MAC.

Finally, we note that Rainbow has the potential to significantly improve applications where reliable data broadcast is required. For example, unmanned aerial vehicles (UAVs) may use Rainbow to distribute surveillance imagery to ground units. In a more sophisticated use case, Rainbow could be used by a UAV to stream surveillance video to ground units participating in a parallelized target discrimination application. That is, Rainbow can play a pivotal role in wireless distributed computing, where it can be considered as a reliable and efficient method for broadcasting the input data of a task to multiple computation nodes. Our work on DWARF clearly indicates the need for such a method. Future work will focus on pursuing this line of investigation.

5.4 Summary

An important overall conclusion is that technologies in wireless sensing, computing and networking can work together to gain substantial performance improvements for wireless systems and applications. For example, for the distributed speaker identification application we have studied, sensor diversity is essential in extracting high-quality speaker features, and network coding is most effective in distributing audio data sets over an ad-hoc wireless network.

6.0 REFERENCES

1. Hubers, P. J., **Robust Statistics**, Wiley, New York, NY, 1981.
2. Shang, Y., Ruml, W., Zhang, Y., Fromherz, M., "Localization from connectivity in sensor networks", *IEEE Transactions on Parallel and Distributed Systems*, **15**, 11, 2004, pp. 961-974.
3. Whitehouse, C., **The Design of Calamari: an Ad-Hoc Localization System for Sensor Networks**, UC Berkeley, Berkeley, CA, 2002.
4. Bachrach, J., Taylor, C., "Localization in Sensor Networks", *Handbook of Sensor Networks: Algorithms and Architectures, 1st ed.*, **1**, 2005.
5. Williams, B., Camp, B., "Comparison of Broadcast Techniques for Mobile Ad Hoc Networks", *Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking & Computing*, 2002, pp. 194-205.
6. Mosse, D., Melhem, R., Ghosh, S., "Analysis of a fault-tolerant multiprocessor scheduling algorithm", *24th International Symposium on Fault-Tolerant Computing*, June 1994.
7. Guerraoui., R., Schiper, A., "Software-Based Replication for Fault Tolerance", *Computer*, **30**, 4, 1997, pp. 68-74.
8. Dean, J., Ghemawat, S., "MapReduce: simplified data processing on large clusters", *Communications of the ACM*, **51**, 1, 2008, pp. 107-113.
9. Burrows, M., "The Chubby lock service for loosely-coupled distributed systems", *Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation*, 2006, pp. 335-350.
10. Ho, C., Obraczka, K., Tsudik, G., Viswanath, K., "Flooding for reliable multicast in multi-hop ad hoc networks", *Wireless Networks*, **7**, 6, 2001, pp. 627-634.
11. Ni, S-Y., Tseng, Y-C., Chen, Y-S., Sheu, J-P., "The broadcast storm problem in a mobile ad hoc network", *Proceedings of the 5th annual ACM/IEEE International Conference on Mobile Computing and Networking*, 1999, pp. 151-162.
12. Lim, H., Kim, C., "Multicast tree construction and flooding in wireless ad hoc networks", *Proceedings of the 3rd ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, 2000, pp. 61-68.
13. Ware, C., Judge, J., Chicharo, J., Dutkiewicz, E., "Unfairness and capture behaviour in 802.11 ad-hoc networks", IEEE International Conference on Communications, **1**, 2000, pp. 159-163.

14. Togabi, F. and Kleinrock, L., “Packet Switching in Radio Channels: Part II--The Hidden Terminal Problem in Carrier Sense Multiple-Access and the Busy Tone Solution”, *IEEE Transactions on Communications*, **23**, 12, Dec 1975, pp. 1417-1433.
15. Chou, P., Wu, Y., Jain, K., “Practical network coding”, *Allerton Conference on Communication, Control, and Computing*, 2003.
16. Mitzenmacher, M., Upfal, E., **Probability and Computing: Randomized Algorithms and Probabilistic Analysis**, Cambridge University Press, Cambridge, UK, 2005, pp. 32-34.
17. Hamra, A., Barakat, C.,Turletti, T., “Network Coding for Wireless Mesh Networks: A Case Study”, *Proceedings of the International Symposium on on World of Wireless, Mobile and Multimedia Networks*, 2006, pp. 103-114.
18. Lee, U., Park, J-S., Yeh, J., Pau, G., Gerla, M., “Code torrent: content distribution using network coding in VANET”, *Proceedings of the 1st International Workshop on Decentralized Resource Sharing in Mobile Computing and Networking*, 2006, pp., 1-5.
19. Widmer, J., Le Boudec, J-Y., “Network coding for efficient communication in extreme networks”, *Proceeding of the 2005 ACM SIGCOMM Workshop on Delay-Tolerant Networking*, 2005.
20. Wang, M. and Li, B., “Lava: A Reality Check of Network Coding in Peer-to-Peer Live Streaming”, *Proceedings of the 26th IEEE International Conference on Computer Communications*, 2007, pp. 1082-1090.
21. Ghaderi, M., and Towsley, D., and Kurose, J., “Network Coding Performance for Reliable Multicast”, *Proceedings of the IEEE Military Communications Conference*, 2007, pp. 1-7.
22. Gkantsidis, C., Rodriguez, P. R., “Network coding for large scale content distribution”, *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, **4**, 2005, pp. 2235-2245.
23. Maymounkov, P., Harvey, N., Lun, D., “Methods for Efficient Network Coding”, *Allerton Conference on Communication, Control, and Computing*, 2006.
24. Aguayo, D., Bicket, J., Biswas, S., Judd, G., Morris, R., “Link-level Measurements from an 802.11b Mesh Network”, *ACM SIGCOMM Computer Communication Review*, **34**, 4, 2004, pp. 121-132.
25. Angkititrakul, P., Hansen., J., “Discriminative in-set/out-of-set speaker recognition”, *IEEE Transactions on Audio, Speech and Language Processing*, **15**, 2007, pp. 498-508.

7.0 LIST OF SYMBOLS, ABBREVIATIONS, AND ACRONYMS

DoD	Department of Defense
DWARF	Distributed Wireless Application Runtime Framework
MAC	medium access control
MDS-MAP	Multi-Dimensional Scaling Map
MDS-MAP(C)	Multi-Dimensional Scaling Map (Classic)
MDS-MAP(C,R)	Multi-Dimensional Scaling Map (Classic, Refined)
NIST	National Institutes of Standards and Technology
OLPC	One Laptop Per Child
RSSI	received signal strength indicator
SID	speaker identification
SISR	Snap-Inducing Shaped Residuals
SNR	signal-to-noise ratio
TIMIT	Texas Instruments/Massachusetts Institute of Technology
UAV	unmanned aerial vehicle